INF-3981

Master's Thesis in Computer Science

Securing Private Peer-To-Peer Networks

Lars A. Fredriksen

AUGUST, 2007

Faculty of Science

Department of Computer Science

University of Tromsø

--

INF-3981

Master's Thesis in Computer Science

SECURING PRIVATE PEER-TO- PEER NETWORKS

Lars A. Fredriksen

August, 2007

--

Table of contents

--

--

--

Abstract

This thesis describes the research of grading security in private Peer-to-Peer (P2P) networks, and ultimately the development of “The Socialized.Net Embedded – Cryptography Version” (Tsnecv). Tsnecv is a revision of “The Socialized.Net Embedded” (Tsne) which is the embedded version of Njål Borch’s doctorate “The Socialized.Net”. Tsne is a P2P file sharing application which builds it private P2P network as an Ad-Hoc or Distributed Transient Network. Tsnecv focuses on applying different levels of security to the network with respect to authentication of peers and access to resources, primarily through the use of public key cryptography and assignment of varying trust to peers that meet in the network.

The goal is to establish secure authenticated communications in such a way that peers may be assigned different policies with respect to access of files and resources, and in this way introduce different levels, or rather grade the security and trust of other peers. An exiting feature is the possibility to use a wireless device to perform a search among the files of all your friends’ and friends’ friends, or other people you have passed by, and automatically having your living room media PC stream the live audio. Files are accessed based on user groups. Someone who forms an Ad-Hoc (spontaneous) network with your wlan unit while passing you by, may autonomously assign you a low trust level, and thus probably access to few or no files. Your close friends however, may grant you access to everything but their most private files.

An important aspect was attempting a transparent integration between Tsne and the new levels of security and the mechanisms used to obtain them. It was attempted to inconvenience the users as little as possible, while keeping the accessibility of available resources as high as possible for all peers, while still allowing as much control as possible. Not only is it important to be able to grade the security at different levels, but it would be nice if users did not have to stop and ask each other for passwords, keys, secrets or to carry memory sticks in case they meet someone new and interesting. In other words, to keep the autonomy as intact as possible and the resources plentiful while allowing peers to control access to their shared resources.

Public (as in asymmetric) key encryption was the choice of tool to achieve authentication of nodes. Web Of Trust was used as a starting point for the exchange of keys, but Tsnecv grades both nodes and networks at different levels of security, so in some cases you may meet someone new and exchange a key autonomously and publicly, but still consider the security of the key and the association of an identity to the key strong enough to securely authenticate a peer at a later time.

Keywords: P2P private peer-to-peer trust security access authentication

--

Acknowledgements

I would like to thank Kristina who was extremely understanding during the last weeks of my writing and kept the household afloat among papers, books, and a continuos flow of boxes with take away. She was also very understanding during my constant monologues where I talked myself through events leading to problems during the development of the multithreaded cryptographic networked computing system .

I would like to thank Anders Andersen at the University in Tromsø, who was willing to be my teaching supervisor. I probably should have requested more advise. The first advice I got was to write down absolutely everything I did along the way, any decision I made, why I made it, and make sure to include even the smallest detail. During the final days when the last strands of powdered coffee were dwindling and my eyes wanted to look at anything but a monitor, these notes helped a lot.

I would like to thank my external teaching supervisor Njål Borch. I would especially like to thank him for the starting point, TheSocialized.Net Embedded, which in spite few comments in the code, was a joy to experiment with.

It has been enjoyable to work with a project where the reward has been a continuous flow of music between my machines. I would therefore like to thank the artist Morten Abel who sang ‘INPARTICULAR’, and the girl who sang ‘JUHUUU’ in the background, every time a test of the system was successful.

I would like to thank all the contributors of music to the Creative Commons project. Without them public testing could have been quite cumbersome.

Finally I would like to thank all the contributors to projects hosted at Sourceforge. The tools and Python extensions I obtained from Sourceforge have been invaluable.

Any and all copyrighted music was of course legally obtained and only made available to myself in a restricted user group of type 8 or 9. In other words, no artist was harmed during the research of this thesis.

Finally, I would like to thank my family for providing the toy phones, and providing me with other means of exploring my interest at a young age.

1. Introduction

1.1. Private Peer-to-Peer networks

Peer-to-Peer (P2P) networks have grown popular due to their inherent scalability and low cost. P2P file sharing networks are to a large extent open, global networks, with very limited or no access limitations.

A specialized form of P2P networks, are private P2P networks. In a private P2P network, only a limited selection of entities are granted access to the system. For peers in some of these private P2P networks it may be desirable to assign groups of selected entities different access restrictions, or policies. An example of a group is a person's home computer, a media PC connected to a television, a handheld unit and a portable computer. In such a closed group, one might desire to share mp3 files, photographs and home movies among the units without making them accessible to strangers. Another form of limited group might be close friends and family, whom a peer wishes to grant access to mp3's, without granting access to other private parts of the files system. There are many aspects to consider in such a system. Among them are: How to include units in the limited group, how to limit access to the system and shared resources, how to protect the exchange against eves dropping, and even the consideration of whether or not it is desirable to protect this exchange .

1.2. Goals

This project will research which possibilities are available to secure private P2P networks. Some known approaches to security in systems with limited access are 'Web of Trust', certification through certificates based on asymmetric public key cryptography (and a trusted third party) and other cryptography based systems. Important aspects of all these systems are confidentiality (access), integrity (authentication), and trust (who can you trust?). Some of the benefits one may achieve is control of to whom data is published to, and guaranties of the integrity of data originating from trusted nodes. For those who are interested in protecting themselves from frivolous copyright infringement lawsuits of the type that is spreading over the world, it might be an important aspect to be capable of securely limiting access to private network to comply with current legislation.

It is the opinion of the author that one of the characteristics which makes P2P networks popular is their autonomy. The peers generally does not need to concern them selves with how other peers connect to the network. The peers simply use the P2P application to access resources while the network grows and handles nearly everything else on its own. One of the goals will thus be to retain much of this autonomy, while still allowing the security to be graded. The question which will be researched is:

How may access to resources shared among peers in private Peer-to-Peer networks be graded, in a secure manner, and with limited intervention by the users.

With the rise of handheld wireless devices, more and more private networks operate in so called disconnected mode, in other words, the networks are not connected to the internet at all times (or maybe not at all). The users also has a much more limited user interface than is normally found on a regular desktop computer. There is often no mouse, only small keys, which makes it important for a system to be able to handle as much as possible of the decision making by it self. This project will thus have as one of its goals, to look at the possibilities of securing private P2P networks, but at the same time be as little, or not at all dependant on centralized servers.

A part of the project will be to further develop an existing P2P system in such a way that the levels of security may be graded. Njål Borch developed the system “The Socialized.Net Embedded” (Tsne). Tsne is an adaptation of his doctorate , “The Socialized.Net”, designed specifically to stream mp3 (music files) between peers in an Ad-Hoc network. The implementation of graded security will build on the analysis of the different possible approaches which will be part One of this project. The implementation will be a specialized form of the mechanisms that are researched for grading the security in private P2P networks. Important aspects will be some of the things that are already mentioned; access policies for limited groups, authentication, and trust. In some instances you trust your friends’ friends, while other times, you might only trust your friends, or maybe just your self and you own systems. In this setting we will explore other implementations, cryptography, which entities are trustworthy, suitable trusted authorities, and if we even need such authorities in our system. The purpose of the developed system will not be to produce a complete and functioning network. The implementation of Tsnecv will be developed to enable experimentation with the different aspects of security, and should enable us to observe how system grades security . The system should have a fairly easy way of manipulating data which defines the security of the system. By doing this, it will hopefully be possible to show how choices that are made influence the security and autonomy.

--

2. Background

2.1. Definitions

As Platoon once expressed, before something may be discussed, the definitions of the topic at hand must be clear. The P2P technology is a relatively new concept with conflicting definitions and ambiguities. In the following subsections the authors usage of important terms will be clarified.

2.1.1. Ad-hoc and Distributed Transient Networks

The term Ad-hoc[3] networks are most commonly used about wireless devices which spontaneously forms networks among them by connecting directly to each other, and usually forming an overlay network. Ad-Hoc is however also a suitable term for nodes on a local area network(LAN)[28]. which spontaneously form an overlay network. Ad-hoc networks are a part of the Distributed Transient Network(DTN) paradigm. The definition of DTN networks is somewhat broader and is defined as: “the type of network which is inherently decentralized by nature and consists mainly of nodes which are not per se constantly a part of the network and are able to join or leave at any time at any place in the network”[2].

2.1.2. Peer

A peer is one of many entities connected to a P2P network with a P2P application. When referring to a peer it will be both the application and inherently the user of the application that is being referred. In pure P2P networks there are only peers. Other P2P networks rely on centralized servers in one form or the other, or relies on concepts such as special peers referred to as super nodes[29,30]. When concepts such as super nodes or servers are discussed, they will not be referred to as peers.

2.1.3. Peer-to-Peer application

P2P applications allows peers to access resources other peers on the same P2P network is sharing, and it allows peers to make resources available to other peers. The resources are generally files which may contain virtually anything. Modern P2P applications and protocols are sometimes more general and may allow the routing of any type of data stream, such as text messages, voice streams, video streams, and generally any other type of communication.

2.1.4. Peer-to-Peer network

A P2P network is considered the entire system that allows peers to participate in their sharing activity. It is the P2P application itself, the physical network that allows the machines running the applications to connect, and all (if any) specialized applications that may be acting as some sort of centralized server.

Some of these P2P networks rely heavily on centralized servers (even though they may be replicated) for locating other peers and / or to perform searches or even transfer files, especially when problems occur[1]. Other systems are largely decentralized and it is usually the peers themselves that keep track of and inform each other of the addresses of other participating peers. Such P2P networks are generally referred to as overlay networks. This is because searches and sometimes transfers follow a path routed among the peers, and generally do not need an intermediate server for its basic operation. Still, these systems relies on some sort of mechanism to allow new peers into the P2P network for the first time. This mechanism may be as simple as posting the IP address of a peer in the network on a website.

2.1.5. Private Peer-to-Peer network

In sections where only private P2P networks are being discussed, the wording private P2P network will normally only be used the first time it occurs unless it is probable that this could cause ambiguity. After that it will only be referred to as network.

Private implies that the network is only available to selected peers. There are many aspects of privacy, but in this setting it is used to describe a network where only a selected group of peers are allowed access to the network and its resources. Nothing is implied about how this group is selected, the anonymity of the peers or the confidentiality of transferred resources, and most definitely nothing is implied about attempts to conceal that something is being transferred.

The distinguishing characteristics between P2P networks and more traditional networking applications are diminishing and it is becoming harder to distinguish a P2P network from applications like Windows Live Messenger, some web sites, and even new concepts for advanced web browsers. To alleviate this somewhat we require that the resources and infrastructures of private P2P networks are provided by the peers. We also demand that there is

---

[1] One such problem could arise when two peers are connected through a natural address translation device. To avoid using some form of third party, the NAT devices would usually need to be manually configured for incoming traffic to reach the correct host system. Even then only one peer could be connected to each NAT device (assuming the P2P application relies on the use of default ports).

some level of trust involved when a new peer is granted or gains access to the network (otherwise one might possibly argue that it is only a cumbersome P2P network).

As an example, the P2P network Direct Connect (DC) [8,9,10,14] is used both as a regular P2P network and as a private P2P network. There are a multitude of DC networks scattered around the Internet and the different DC P2P networks entry points are centralized servers (hubs). A peer connects to a DC network by contacting[2] one of these hubs, and is immediately ready to participate in file sharing and chat rooms. The only difference between a DC P2P network and a DC private P2P network is that the hub of the private network requires a username and password before a peer is allowed access.

In private networks new peers are usually allowed into the network by an existing trusted peer, and the new peer is usually considered as trustworthy as the existing peer. Because the chain of trust has a tendency to grow over time, this type of private network is generally suited for smaller groups of peers. In some networks new peers will have to build trust over time, maybe by chatting and transferring files, before they gain the same trust as any other peer (and even then they may never gain the same trust). As we will see later on, there are other forms of trust that may be suitable for granting access to a private P2P network. In Tsne the trust is implied in the fact that peers have at one point or another been connected to the same LAN . Nothing is implied about the effectiveness of the security mechanisms many of these private P2P networks use to protect themselves from unauthorized use or access.

The peers in private P2P networks are generally controlled by different users. Even though one user may have a few peers connected at different locations or different times, private P2P networks are suitable and intended for connections between different users. This generally excludes the concept of sensor networks[31] because these networks contain nodes controlled by a single user or organization operating completely autonomously. Sensor networks are not referred to as private P2P networks in this thesis. The security concepts discussed may or may not be applicable for sensor networks.

2.2. Background

The author has always had a special interest in security systems. While enrolled in elementary school (ages 6 through 13) he appeared in a radio

[2] Selections of many public hub addresses are hosted on web servers that are pre configured id the DC client. A DC client may select one or several of these hubs (depending on implementation), or connect to hubs with addresses the user has obtained manually.

program for children on one of two stations covering the entire nation[3]. The program had a weekly spot to showcase the ideas of young innovators. In the program he demonstrated an alarm system made of toy phones which were wired together with switches and sensors to raise an alarm in different rooms of the house if his room was entered or if his drawers were opened.

This interest was carried onto his interest for computer systems. One of the first programs that was developed (on a Commodore Vic 20 computer with built in support for Commodores implementation of the Basic programming language) was a simple ciphering and deciphering program. It was not sophisticated in any way, but it enabled him to store secret messages on his computer, without the risk of the message being readable by anyone without the secret key used for encryption.

After many years the author was introduced to the wonderful world of the Internet. Few commercial businesses had begun to use The World Wide Web (www) as a sales channel and the Internet as a whole was still, to a large extent, reserved for academic institutions and people with a special interest in computer science. This is how he entered the realm of Internet Relay Chat (IRC) clients, File Transport Protocol (FTP), electronic mail (email) and similar technologies.

It was immediately apparent to the somewhat 'paranoid' author that security was a problem. Even defining security was problematic, but it seemed to have similar properties with the protection of a room with an alarm made from a toy phone, or limiting access to files, or at least limiting who could decipher the content. On a regular basis someone was able to hack into the servers of the local Internet provider (i.e. breach the security) and brag about reading the contents of emails or listening in on IRC conversations. FTP servers which were made available to friends continuously got visits from users who had persuaded someone to give them the passwords, and even a few who had obtained the password by performing a brute force attack[4].

The number of networking applications grew at a fast pace in the following years. Aside from the WWW, email systems, and a few others, a particular type of application rapidly gained in popularity among users, namely file sharing applications. A multitude of applications of this type exists, but during the last years it is the Peer-to-Peer (P2P) applications that have become

[3] The program was a weekly program for children called 'Barnetimen' (which would loosely translate as 'The hour for children') and was broadcast on The Public Broadcasting Services of Norway.

[4] A brute force attack is an attack where a certain combination is needed (like a password) and the combination is found by exhaustively trying combinations until the correct combination is found.

dominant. In P2P applications the users (generally referred to as peers[5]) make files of interest available and participate in a cooperative effort to exchange the files effectively (some applications more successfull than others). P2P networks has many resemblances with the WWW. It is most often files that are moved across a network, and the exchange is generally open for all who wants to participate. As the use of the WWW has broadened, security needs has prompted solutions such as requiring login with passwords to restrict access, the Secure Socket Layer protocol combined with certificates to provide authenticated and confidential communication, and many other innovative solutions. However, in the extensive list of P2P systems the author has experimented with, it seems as if the original intent of P2P, the cooperative and effective distribution of content to all participants, has somewhat clouded other needs and other uses for this technology.

One of the ingenious concepts of P2P is the autonomy. As soon as the application is started, the peer has access to resources. Other peers come and go, and with no interaction from the users standpoint all the shared resources are available. But, it is the opinion of the author that many, if not most or all, users have files they consider private and do not want to make available to all other users of the P2P network they are part of. However, it isn't improbable that these same users might probably want to have remote access to these files themselves, or maybe even grant a few selected peers access to some restricted files. To illustrate this with an example we imagine the P2P system 'Fastswap'. In this system peers are joining and leaving the network without the need of asking other peers to accept them, it seems completely open just like a regular P2P network. However, friends and acquaintances may grant each other access to files they do not want the rest of the network to see. In the same system, a peer may own several machines and be connected to the network from different locations (and appearing in the network as several peers, one for each machine running Fastswap). These users may select to grant special access to even more restricted files, their private files, only to those peers they are running themselves. Maybe they even use the Fastswap application to make other resources than files available.

One small step in solving such problems is a private P2P network. A private P2P network distinguishes itself from a P2P network mainly by limiting the user base and moving the responsibility for approving peers to either the currently participating peers or a peer that is considered trusted. As stated it is

[5] It is observed that in most contexts the distinction between the person who is using a P2P application, the application itself or even the machine or network it is connected to is only vaguely stated. Throughout this thesis a peer will generally be considered an instance of the application executing on a single machine with access to a network adapter and a network. It will not be considered important who is using the application (be it a human or some sort of artificial intelligence). Only in selected sections where identity and authentication is discussed will a distinction be stated.

only a small step, but the basic assumption is that in such a network the peers have some sort of relation with each other, they are not simply random peers scattered around the virtual community of the Internet, and the peers are somewhat stronger associated with an identity. Still, a more fine grained control may be desirable. If this could be combined with the advantages of the P2P technology, a new generation of P2P networks may begin to emerge. If supply is a result of demand, the recent increase in the number of projects in this field and the constant flow of advertisements promising secure P2P services, this should be enough incentive for anyone with an interest to pick up the torch.

The imagined Fastswap application gives rise to a need for security measures. How may users be distinguished from each other ? How may one peer be identified and authenticated by another peer ? How may malicious peers or other entities be restricted from accessing the resources ? How is access to different resources granted to different peers ? How may the autonomy of the P2P application be kept as intact as possible ? These are aspects of different levels of security that may be desirable in a given system. The developed test system will be used to explore these questions, with a focus on graded access, authentication and trust.

3. Problem

Several approaches for securing private P2P networks have been presented by researchers. Some of these ideas have been implemented and some implementations have been based on the developers own ideas. The most striking feature of all of these is their diversity. A small network may find a simple solution where peers connect directly to a centralized server favorable. A single peer may have the responsibility of approving new peers, distributing keys, and evicting abusers. Larger networks may favor a more distributed approach where peers are allowed, at their own discretion, to invite new peers they trust to the network. In other networks it may be some other form of relation, such as location or presence, that allows users into the network, one such system is Tsne. This diverse selection of systems and the diversity of peers using them offers and demands different aspects security. Before we are able to identify the complete scope of the problem, we must look at the different levels of security that may be implemented and desirable in such systems.

3.1. Levels of security

3.1.1. Confidentiality

Confidentiality is :"the concealment of information or resources"[5:pp4]. Access control is one of the mechanisms which is used in computer systems to achieve confidentiality. While information is moving from one system to another over an unsecured channel, additional steps must be taken to retain this confidentiality. The methods used to achieve this generally involves cryptography. Another aspect of confidentiality is the concealment of the existence of resources (resource hiding) or transmissions of these resources.

3.1.2. Integrity

Data integrity and origin integrity refers to the trustworthiness of resources[5]. For data to have integrity it must be certain that it has not been subject to unauthorized change. Data integrity applies to both stored and transmitted data. Origin integrity is more commonly referred to as authentication. Authentication applies to the source of data, and cryptographic mechanisms may be used to enable the receiver of data to be certain who the original source of the data was. In the setting of this thesis, where peers are generally communicating through the public Internet, it is not the prevention of unauthorized changes of transmissions that is important, but the detection of such changes.

3.1.3. Availability

"Availability refers to the ability to use the information or resources desired"[5:pp6]. For a system to be reliable it should be resilient against actions or behaviors which deliberately or even inadvertently makes resources that are supposed to be available inaccessible.

3.2. Threats

Threats are possible ways to attack a system. If the attack is successful it may lead to use which violates the intended security[5]. Successful attacks breaches the policies for one or more of the discussed security services, confidentiality, integrity and availability. Successful attacks may result in some widely recognized problems [5:pp7]:

- Snooping
- Masquerading or spoofing
- Delay
- Denial of Service
- Unauthorized modification or alteration

--

- Repudiation of origin
- Denial of receipt

3.3. Assumptions, trust and assurance

Systems which implement some level of security rely on the concepts trust and assurance. Systems are never stronger than their weakest link and it is important to identify which components (be it hardware, software, human or other) represent a threat to this security. If a system is considered secure because it has locked gate, the security may be breached if unauthorized personnel has access to the key, or if the door is made of paper. If however all personnel who has access to the key is considered trusted and the door is considered impenetrable, and these assumption are correct, the systems security may not be breached. Of course this relies on a specification of the security which states that the locked door should only be unlocked by authorized personnel. If the intention is to keep the room behind the door secure, the specification would also have to state that the room should only be possible to enter through the unlocked door (and thus steps would have to be taken to make this the only possible way to enter). This means that for the intention of a security policy to be achieved, a specification which implementation prevents all possible threats from breaching the security must be stated. Because of this the security of a system relies on the assumption that the specification is correct and includes all possible ways to breach the system.

3.4. Goals of security

According to Bishop[5], the proper specification of a given security policy's secure and non secure actions or states allows for the implementation of mechanisms which may prevent, detect or recover from attacks. When designing a secure system it must be decided which of these mechanisms are needed. For a home computer with no private files where the only security policy involves availability of the files, a routinely complete backup routine for the system (and procedures for restoring from backup) may be considered a sufficient specification of the security policy. In most systems the specification is much more complex. In private P2P systems, private is an important indicator of what the specification of the security policy should have as its focus.

3.5. Problem to solve

3.5.1. Authentication

3.5.1.1. Authenticate

The technology that is being researched is private P2P networks, and we are looking at mechanisms for grading the security. Following the definition of a private P2P network in section 2.1.5 an important characteristic is the limitation with respect to which peers are allowed access. This is part of the integrity security, specifically origin integrity, more commonly referred to as authentication. The first problem that must be solved is a specification of a mechanism for a peer to be authenticated before it is allowed access to the network. It would be preferable if this authentication was capable of grading new peers according to the level of trust existing peers have in them. This differentiation should attempt to be relatively autonomous in order to retain the autonomy which we must presume is at least partly responsible for the prosperous spread of the P2P technology.

3.5.1.2. Identity

As explained, peers joining private P2P networks are generally invited by peers who are already part of the network. When the joining peer connects and the network attempts to authenticate it, the networks attempts to authenticate the connecting machine, or at least the origin of the data which is being transmitted. Identity and authentication is not the same. Identity may in the scope of a private P2P network be described as the glue between the authenticated node and the user controlling it. The identity doesn't necessarily identify a person by a name or other personal information, but the identity allows others peers to at least differentiate between one peer and another. It will be attempted to include some sort of applicable identity as a part of the authentication mechanism.

3.5.2. Access

There would be little sense in grading peers during authentication if this grading wasn't applied to resources. Access is a part of confidentiality. This second problem which needs solving is the specification of a mechanism which makes resources accessible only to authenticated peers. If the thrust of an authenticated peers is graded, the access to resources should also be graded accordingly. As with the authentication, the grading of access should be as autonomous as possible to retain the autonomy of the overall system.

3.5.3. A note about autonomy versus control

Autonomy and control are largely opposite characteristics. In section 3.5.1 we made the assumption that the autonomy of P2P networks is a contributing factor in their wide spread use. If this assumption is correct it follows that with two identical systems only differing in the amount of manual operations involved in accepting new peers, the one with the fewest operations is the superior one (at least from the peers viewpoint). However, this thesis researches private P2P networks, which differs from regular P2P networks mainly by the control of who is granted access. Balancing autonomy and control implies a tradeoff where the control comes at a cost of reduced autonomy. The researcher will do his best to limit the reduction of the autonomy.

3.5.4. Focus

Among the aspects of security that are not the focus of this thesis, even though some of them will be discussed briefly, we mention some important ones: concealment, data integrity and availability. From this it follows that among the mentioned threats the concepts delay, denial of service, unauthorized modification or alteration (of stored data), repudiation of origin and denial of receipt are considered less important than snooping, masquerading and spoofing. This thesis will rather focus on the concepts of access (a part of confidentiality) and authentication (the origin integrity part of integrity). Access and authentication are considered important because they are very basic to the operation of keeping networked communication private.

3.5.5. Motivation

A multitude of projects related to P2P networks are under constant development and new projects and ideas are presented. In the authors opinion none of these have sufficiently fused the autonomy of the regular P2P networks with the control associated with private P2P networks. It seems as if the focus of private P2P networks is either networks that hides the origin of data (like Freenet[19,20] where the origin of resources is concealed, or the more common private P2P networks like those based on the Direct Connect technology [8,9,10,14] or Torrent networks where all resources are available to all peers and the invitation to enter is only one of many obstacles. This was the motivation that led the researcher to the task of researching a technology which will hopefully make the development of an application with the control of a private P2P network but the autonomy, broad user base, and resource availability of a regular P2P network possible, or at least bring the two branches of the P2P technology a few steps closer together. From the authors view point, grading in the authentication process of peers and grading of access, both in more autonomous fashions than the current technology is capable of, are the first steps towards a diminishing gap between these two branches of the P2P technology.

4. Related work and technologies

4.1. Security approach for network applications in a Peer-To-Peer perspective

With the capabilities of many current communication devices, spontaneously creating networks (Ad-Hoc networks) or other networks with transient properties, using P2P technologies get a broader application. When someone walks around with their mobile device with its Wireless Local Area Network (WLAN) or Bluetooth turned on, the device will meet many other devices. If a device can control another or access its resources, it becomes obvious that the level of security or policies in many cases should be different. For example the security needs are different when a mobile meets the media PC of its owner, than when the mobile meets 'Joe Shmoe' in the street or while connected to the Internet. Private P2P networks has some of the same characteristics as walking around with a network capable device. Some of the peers you meet are people the user knows, while others may be complete strangers, included in the network by a friend of a friend, or someone along a chain mixed of acquaintances and strangers. In this chapter we will discuss how current technologies has attempted to solve the challenges of the different levels of security discussed in the previous chapter. The chapter is divided in two parts. The first discusses security mechanisms such as passwords and cryptography. The second discusses how these security mechanism are used in the current private P2P technologies and which levels of security the mechanisms are used to achieve.

4.2. Security mechanisms

As discussed in chapter 3, security is based on three concepts, confidentiality, integrity and availability. As stated in section 3.5.4, the focus of security in this thesis is on the concepts of access (a part of confidentiality) and authentication (the origin integrity part of integrity). The following describes technologies for providing these levels of security. Only concepts that are considered somewhat effective will be discussed. For example, controlling access by not publicly distributing the IP address or hostname of a centralized server will not be discussed as access control.

4.2.1. Basic access and authentication mechanisms

Private P2P networks grants access to selected peers. The network uses an authentication mechanism to authenticate these selected peer and then grants access to some or all resources in the network.

4.2.1.1. Secrets as authentication

An unsophisticated system may use the common approach to access of simply requiring that a peer has the knowledge of a secret, like a password. This secret may be a shared password for all selected peers (identifying the peer as belonging to the group of selected peers), or a specific password for this exact peer (identifying the peer as a specific peer belonging to the group of selected peers). If the system is more advanced it will use encryption mechanisms to prevent attackers from snooping the password and masquerading or spoofing the identity of the peer. In both cases a routine for the initial assignment and exchange of the password (shared or specific) must be in place.

4.2.1.2. Locality as a form of authentication

At least one private P2P network relies on locality for authentication. A system like Tsne (discussed in the next chapter) grants initial access to peers which are present on the same LAN, and the network is thus considered private for peers on this LAN (even if the LAN is connected to the Internet). Tsne allows peers which has once met on a LAN to connect to each other through the Internet at a later time. At that point Tsne authenticates peers only by their supplied node ID, which makes spoofing a relatively trivial attack on the system[6].

4.2.2. Cryptographic mechanisms for access and authentication

Private P2P networks aiming for higher levels of security uses advanced cryptographic concepts to achieve this security. An advanced cryptographic system is able to authenticate nodes in such a way that the success of an attack on the cryptographic part of the system becomes infeasible with the resources that are available.

When the public keys of entities are known, cryptographic systems like RSA may be used to authenticate and provide secure communication [6,7]. However, in systems where keys are not previously known, two devices or systems which haven't meet before has the problem of how to exchange their first keys securely. In an encryption system where authentication is a necessity, common solutions include manual distribution of the keys (which are linked to some form of identity), and exchange aided by a trusted third party, sometimes combined with certificate signature schemes.

4.2.2.1. Manual exchange

Manual exchange is often referred to as out of band exchange. This means that the encryption system relies on the keys to be distributed through a separate

[6] There are many other challenges related to access and authentication in Tsne, but these will be described in detail in the design and implementation chapters.

mechanism. This includes solutions like delivering the key by phone, mail, email, floppy disks, other portable storage devices, and several others. It is noted that for the security of the system to be kept intact, the confidentiality of the key itself must be assured.

4.2.2.2. Trusted key servers

Trusted key servers stores the keys of all or a large group of the peers in the network. For a peer to acquire the key of another peer, it only needs to know the key or keys of the trusted key servers. This alleviates the problem of distributing keys, but introduces the problem of a single point of failure in the trusted key servers. For a private P2P network that already relies on centralized servers this is not necessarily an added disadvantage, but for pure P2P systems a centralized trusted key server will add an unnecessary point of failure, affect scalability, and also the need for administration of the server.

4.2.2.3. Certificate Authorities

Another solution is the use of public key cryptography based digital certificate servers, also known as certificate authorities (CAs)[6]. The CA has at least one known public key, which is distributed with applications that has a need for secure communication. Using public key cryptography like RSA and Elliptic Curve Cryptography, the CA creates a certificate for a device (like a web site or a peer) that needs to identify itself. The simplified explanation of the process involved when generating a certificate would be to take the device's public key, some additional info like a Universal Resource Locator (URL) of a web site or some sort of naming representing a peer, packing it together, hashing it, and finally signing the hash with the CA's private key[6]. The certificate may be used to establish both a secure communication channel between peers who have never met or exchanged keys, and to authenticate peers who is attempting to access a private P2P network. The advantage of certificates compared to a trusted key server, is that the peers themselves exchange the certificates, it is only the creation of the certificate that must be done by the centralized servers.

4.2.2.4. Web of Trust

In a web of trust friends have established authenticated communication channels. Peers trust their friends or friends of friends (and so on) to supply the proper key and information about people their friends have the means to establish a secure connection with, but whom the peer does not have a key or certificate for. In its simplest form, a web of trust is used only to supply the a peer with the needed information to establish a secure connection with a friend of the friend. In this type of network there are only two levels of trust. A peer trusts their friend, and the peer selects to trust any friend of the friend because their friend trust the friend of the friend. In some specifications and implementations, intricate webs are created by chains of trust spanning many levels of friends of friends of friends and so on. Because a security system is only as strong as its weakest link, shorter chains of trust are arguably more secure. However, longer chains of trust increase the number of peers that may connect securely without some form of out of band key exchange.

4.2.2.4.1. Web Of Trust in PGP

In PGP a web of trust (WOT)[32,33] provides authentication and secure communication based on public key cryptography and certificates from a network of trust. The WOT does not depend on a central CA, even though WOTs often have several data bases where many certificates are stored. In a WOT devices that meet use their keys to sign each other's certificates. They also add each others list of certificates they trust to their lists of trusted certificates. Gradually as a certificate gets more signatures and the certificate propagates to more devices that trust this certificate, the WOT grows. It is the intention that a route to a destination should always be possible to find. By route we mean a way for two devices who want to communicate, but haven't signed each others certificates, to find a combination of certificates with their associated lists of certificates they trust, which will prove that the two respective certificates of the devices are legitimate. The web of trust is built on the foundation that a device will most often know a device which knows a device who knows a device (and so one), which finally knows a device which has meet and trusted the device with whom one wants to communicate. Before a choice is made to use a certificate scheme like the one in used in PGP or a simpler version, the tradeoff between the increased number of devices that may be authenticated and the lessened security resulting from a long chain should be considered.

4.3. Cryptographic algorithms and protocols

Cryptographic algorithms and protocols form the basis for modern secure networking applications. Cryptography is a broad and advanced field which involves many advanced topics including mathematics sometime beyond the comprehension of most computer scientist. Proving the correctness of an algorithm or protocol and the associated assumptions that are made when specifying their security often requires years of work and the combined effort of hundreds of researchers. This thesis does not focus on the inner workings of cryptographic algorithms, but will in the following introduce some of the important concepts, particularly cryptographic algorithms that are in broad use today and details related to protocols that are relevant to authentication.

4.3.1. Symmetric cryptography

Algorithms for symmetric cryptography use the same key for encryption and decryption. All current renowned symmetric algorithms use block ciphers to encrypt and decrypt. Block ciphers only encrypt one block of a message at a time, consuming a fixed number of bits for each block (with 128 bits being the block size of current block ciphers). According to Ferguson and Schneier[6] a block cipher may be visualized as a very big key-dependent table. For every key there would be a table that map the plaintext to the ciphertext. For a block cipher with 128 bits this table would contain $5 \cdot 10^{39}$ bytes of data. The highest achievable bit security of a n-bit block cipher is the limit n bits. All ciphers have limitations and vulnerabilities (such as certain keys which are considered weak because the allow certain types of statistical attacks and thus are excluded) which reduce the actual bit security slightly.

4.3.1.1. Advanced Encryption Standard

The block cipher 'Data Encryption Standard' (DES) has long been an encryption standard but the encryption is easily broken with current resources[6]. The United States (U.S.) government uses the 'Advanced Encryption Standard'(AES)[6] which is considered secure in years to come[6]. Other block ciphers (like Serpent) which are considered more secure (by most serious cryptographers) exist[6], but due to other factors such as speed, ease of implementation and the strong position of the U.S. government and trust in their evaluation of the AES specification, AES is the de-facto standard. One large advantage of using AES for symmetric encryption is the fact that it has undergone a huge amount of scrutiny and analysis.

4.3.2. Asymmetric cryptography

Algorithms for asymmetric cryptography, also known as public-key encryption, use two separate keys for encryption and decryption. In its general use, the private key is used to encrypt a message or cleartext to ciphertext, and the public key is used to decrypt the ciphertext to cleartext. The current algorithms are mathematical calculations where knowledge of the public key reveals nothing of the private key. Some of the algorithms may be used to provide both digital signatures and public-key encryption. Digital signatures are very useful for systems that has a need for authentication.

4.3.2.1. RSA

The RSA[23] algorithm, named after its inventors Ronald Rivest, Adi Shamir and Leonard Adleman, was published in 1978. According to Ferguson and Schneier[6] RSA relies on a mathematical concept known as a Trapdoor One Way Function. Johnsen[7] explains the security of RSA in terms of the complexity of factoring large primes and modular multiplication. In an RSA system an entity uses the mathematical algorithm to generate a key pair consisting of a public and a private key, and the public key is published. Anyone may use this key to encrypt cleartext to ciphertext. Even though everyone has access to the public key, only the entity which produced the private key will be able to decrypt the ciphertext to cleartext. This allows anyone to produce encrypted messages which may only be decrypted by the desired entity (given that the authenticity and data integrity of the public key is secure). If the private key is used to encrypt a cleartext, decryption with the public key will result in the cleartext. It may seem of little interest to encrypt a message in such a way that everyone (because everyone has access to the public key) may decrypt it. However, this property allows RSA to be used in a signature scheme, and allows the owner of the private key to sign messages, which in turn allows recipients to confirm the authenticity of the sender.

A strong competitor for the RSA system is Eliptic Curve Cryptography(ECC)[34]. ECC is probably secure or even more secure than RSA (assuming of course that for both systems the keys that are said to produce the same calculated bit security is used). The advantage of ECC is that the increase in the length of the key required to increase its bit security, grows much slower than it does for RSA. RSA is considered safe for many years to come, but the U.S. government has already switched to ECC (which means cryptographic systems that are purchased by them must use ECC not RSA). The problem with RSA is that it will not be possible to produce keys of sufficient length some time in the foreseeable future. Because the required keys for similar bit security in ECC are much shorter, ECC will outlast RSA.

For developers there is sometimes no choice to be made. ECC is a new technology, and unless the developer intends to implement the internals of the cryptographic parts of the system, RSA is often the only available (of the two competitors) cryptographic solution for the development platform. There are of course other public-key solutions, but RSA is probably the most widely used and best known of them all[6]. As with AES, RSA has undergone a huge amount of scrutiny and analysis after it was published.

4.3.3. Theory of Authentication protocols

Authentication protocols allow entities to confirm the origin integrity of data. In a private P2P network, public-key cryptography may be used to both authenticate an entity (e.g. a peer) and to establish a secure channel for communication over an unsecured network.

There are several approaches to the implementation of authentication using public-key cryptography and signatures. Ferguson and Schneier[6] explains RSA signatures. Current hashing algorithms are much faster than encryption with RSA. Hashing a message and signing only the hash has both performance and security advantages. In addition RSA can only encrypt or sign bitstrings that are shorter than its key (actually key length in bits minus 1), so signing a message which was longer would require some sort of mechanism similar to cipher block chaining[6] to split the message into sequences of proper length.

--

Using the commonly known abstractions of entities in the cryptographic world, Bob, Alice and Eve, an authentication could follow the following sequence:

Bob   Alice

Enc$_{AlicePub}$(

Sign(Hash(I'm Bob,Bob'sChallenge,SymKey),

(I'm Bob,Bob'sChallenge,SymKey))   ->

<-   Enc$_{BobPub}$ (

Sign (Hash(I'm Alice,Alice'sChallenge,BobChallengeResponse),

(I'm Alice,Alice'sChallenge,BobChallengeResponse))

EncAlicePub(

Sign(AliceChallengeResponse),

(AliceChallengeResponse)) ->

"I'm Bob" or "I'm Alice" are expressions of some sort of permanent and usually unique identifier for a peer. Bob and Alice hash and then encrypt the hash of the challenges, challenge responses, identities and the symmetric key for future communication. When the messages are received and decrypted, the receiver hashes the information shown on the last line in each stage above. The signature on the first line in each stage is then verified against the produced hash and the public key of the sender. If any part of the message has been changed by an adversary, the signature verification will fail.

Eve has the ability to snoop and change the contents of exchanged messages. Because the messages are encrypted with the public keys of the recipients, only the intended receiver may decrypt the challenge and give the correct response. If the challenge is answered correctly Bob has authenticated Alice, and vice versa. If the symmetric session key supplied by Bob is used for further communications, the content of the communication will be secret from anyone else than Bob and Alice. Cryptographic systems are intended to secure in different ways. Depending of the specification of the security the exact protocols may have important differences. For example, it is usual to use some form of mechanism to defeat replays of messages. For example, part of the encrypted message may be a sequence number or a time stamp. Handling of out of sequence or untimely messages depend completely on the demands of the systems security. A very useful part of properly authenticating peers and handling exploits such as replays, is that attackers or malicious peers are unable to spoof the user identities of offline peers to gain access to resources and privileges intended for the spoofed peer. If the exchanged session key is used for communication, even if an attacker has acted as a man in the middle, the only thing which is revealed is the presence of the communication, not the meaning of its contents.

4.4. Grading of security in current private P2P networks

In this section we explore different private P2P networks to discover if the security of the network is graded, and how this is achieved. As our definition of a private P2P network states that access is granted to selected peers, any and all networks we discuss has at least graded access security at two levels. Either a peer is granted access, or the access is denied. It is not necessarily how secure the network is that is explored. It is rather the ability of the network to use differentiated levels of security for both access and authentication.

The grading of the security which is to be designed in the next chapter will be done in an existing system which is a private pure P2P network, i.e. it has no centralized servers. The existing technology in these types of networks will be of special intrest. The very recent survey of private P2P networks by Roger and Bhatti [1] is an informative starting point for information about such networks. A representative selection of current private P2P networks such as those surveyed here will be presented in the following. The networks will however be analyzed with respect to grading of security instead of topologies, search routines, discovery, etc.

4.4.1. Direct Connect

In the direct connect (DC)[8,9,10,14] network, peers connect to a centralized server referred to as a hub. Searching is done by querying the hub which stores the file lists of all online peers. When a peer requests a file it is redirected to the peer that is hosting the given file, and downloads it directly. Direct connect is more a protocol than a single application. Many versions and variations of implementations exists. Aside from the original DC system by NeoModus, which spawned the reverse engineered version ‘open direct connect’, the probably most well known is DC++[10]. DC++ allows peers to connect to several hubs (and thus networks) at the same time. In the DC communities the hubs also serves a chat room where peers may socialize, and through the hub users may also create private chat rooms.

In its original form, the servers were controlled by one peer and there were no restrictions to access the network. With newer versions of the hubs like YnHub[11], private networks are created by requiring a password from connecting peers, normally one password per peer. The key is exchanged out of band. There is no cryptographic authentication of the peer. This makes spoofing of user identities a real threat for private DC networks.

--

4.4.2. Octopod

Octopod[12] is in some ways the least autonomous and most restrictive of the discusses technologies. The system consists of peers running the same application. Keys are exchanged automatically through a distributed hash table[13](DHT), but must sometimes be exchanged manually. Each peer creates one or several share directories. For any peer to be granted access to this share, the sharing peer must manually add the peer who is being granted access to the share. In addition, for a peer who is granted access to be able to access the files in the shared folder, a share ID must be delivered out of band from the sharing peer to any peer which is granted access. The Octopod network is probably secure, but the cumbersome addition of peers to shares will make the growth of the network slow, and the extremely restricted environment may arguably be closer related to group ware than a regular private P2P network. The access may however be graded at any level. A sharing peer may add all peers to a single share, in addition to granting access to specific shares to a few or one single peer. It is not possible to add all peers to a single share automatically.

4.4.3. Waste

The WASTE[15,16] networks consists of several completely separate groups of peers running the same application. In each network (group) peers use their group ID in combination with their user ID when connecting other peers to avoid inadvertent joining of the different networks (traffic from foreign group IDs are ignored). The connections between peers are encrypted using a common key. Communication between the peers is secure, granted that none of the peers disclose the key to a third party. There is no method of authenticating individual peers. The IP address of a connected peer is required to connect to the network the first time, but IP addresses of participated peers are propagated through the network once a peer connects to the network, and the IP address of peers are updated when they connect from new addresses. Even though the initial connection procedure is somewhat advanced, the further use of the system is simple. Files may be pushed to specific peers, and shared files of other peers may be browsed and downloaded. Except from the manual pushing of files there is no grading of access, nor authentication of peers. Even though the connections between peers are encrypted, each intermediate peer may decrypt the passed messages, allowing snooping and spoofing.

4.4.4. Retroshare

Retroshare[17] (and several similar systems which are omitted because they rely on a web service controlled by unrelated third parties) uses cryptography to establish secure channels between peers who are friends. Files are transmitted directly between friends, or through an intermediary common friend when strangers communicate (and only if such a common friend exist).

Retroshare does not rely on a centralized server in the control of a third party. Retroshare achieves this through the use of a DHT. The public keys of a peer is packed in a certificate. This certificate is then signed by friends. The signed certificates are distributed through the DHT, and as the number of signatures on each certificate increases, it becomes more likely that a joining peer will only have to obtain a few signatures out of band before it may connect securely to most of its friends. Certificates that are signed by a friend (who is supposed to have established the identity of the certificate owner) may be signed in band. The certificate/signature scheme is a variation of a web of trust (discusses earlier in this chapter).

Friends are generally peers who have some social connection. Peers who are not friends but share a common friend may communicate through the secure channel that is created by their two separate secure channels to the common friend acting as an intermediary peer. These types of private P2P networks are from the average users viewpoint very similar to applications such as the MSN messenger.

The Retroshare documentation is ambiguous. At one point it states that only communication between friends is safe, while at another it states that files are only visible and accessible to friends and to friends of friends who are granted access through the auto discovery mechanism. The ambiguity probably has to do with the Retroshare team's definition of secure. The access and visibility of the files are only available to friends and friends of friends, but it is not considered "secure" to transfer a file to or from a friend of a friend. It is assumed that they intend to express that if you transfer something to someone you don't know, it is not always wise to trust your friends judgement of the intentions of this friend of your friend. Grading of access is very limited. Files are either shared or not shared. If a peer is considered a friend, access is gained to all files. If a friend of a friend connects through the auto discovery mechanism, the sharing peers must either grant access to all or no files. Only strangers who are friends with a common friend may connect through the auto discovery mechanism.

--

4.4.5. Turtle

Turtle[18] uses password mechanisms to generate and distribute keys. In Turtle friends who want to authenticate for the first time use out of band communication to agree on a question they both know the answer to. The answer, which is not revealed or exchanged, is used to generate an encryption key. This enables the two friends to authenticate each other securely. The use of a public question with a secret answer based on a social relation may increase the security, because snooping of the out of band communication will not automatically reveal the key. The opposite may also be true. Because users will trust the scheme, they may be more likely to use channels that are less secure than they normally would to agree on a question. Where a physical exchange may normally be used, regular email or a chat client may be used when users rely on the concept. This could decrease the security, especially if attackers have a social relationship with the peers, because they are more likely to make educated guesses (or even know) of the answer to the publicly discussed question.

4.4.6. Freenet

Freenet[19,20] is not a private P2P network by it self. Freenet uses a combination of signatures of data and signatures of pseudonyms of the peer which inject data. The Freenet network effectively conceals the origin of data, and only allows a peer which holds the private key of the signatures to inject data under the given pseudonym. The injected data is distributed among the peers in the DHT according to a stochastic algorithm. Peers in the network are not able to determine if a file is being injected in the part of the DHT they are responsible for, or if the data is simply being redistributed to increase the availability. It is noted by the author that because the network is completely open, attackers may use multiple identities and thus there is the possibility that a statistical attack to reveal publishers of data may exist.

To use Freenet as a private P2P network, the peers agree, out of band, on a pseudonym to use when injecting data. They also share the knowledge of the private key. To keep access security, the data must be encrypted before injecting it in the network (otherwise all peers may access it), so an additional encryption key must also be shared in the private P2P network. To clarify, any peer may access all files in a Freenet network, but peers who want to use this same network as a private network, takes some additional steps. They encrypt the data before they inject it, they use a key they have agreed on, and they have also shared the knowledge required to publish data under a pseudonym, and this pseudonym identifies the data they are going to share among them.

The Freenet is considered effective in its concealment the origin of data. Groups of peers who want to use it as a private P2P network basically have to use some sort of out of band communication to distribute the secret keys to new peers who are invited. If the keys are leaked at any time, all data will probably be available for download from the network for a long time after the leak, and all the data will be accessible and may be decrypted by anyone with the leaked key. For sharing where the security only depends on concealing the identity this may be acceptable, but in such a system, using Freenet in its original form may be sufficient. If the confidentiality of the data should be keept secure for a limited time (for example to distribute files during a short period without drawing attention), it may still be a good idea to use the described approach for a private P2P Freenet. In such a setting it may be acceptable that someone leaks the key after a while.

The Freenet is innovative with respect to its charter to fight censorship. As a private P2P network it relies on the same principles of out of band key exchange among peers with some form of social relationship. Depending on the intended use by a peer, the solution may be considered very interesting or completely unacceptable. For example, the only way to grade security with respect to access, is to form one group which each set of peers that is supposed to get access to a given set of files. This would imply that all peers in each group intend to share the same 'types' of files with each other. For example, five friends form one group. Two of these friends wants to share some special files without making them available to the others, and must thus create another group. One of these peers wants to share yet another set of special files with another friend in the group, and thus another group must be formed. For each created group, new keys must be exchanged out of band.

4.4.7. Tsne

In Tsne[7] peers gain access to to a network by broadcasting an announce message on a LAN. All peers who receive the announcement adds the peer to its list of active nodes. There is no authentication mechanism, it is the trust peers on the physical network has in each other that is the basis for the private P2P network. When a node moves from the LAN to a different location that is connected to the LAN but not on the LAN (for example to another LAN which is connected to the first LAN through the internet), it will attempt to connect with the nodes on the original LAN based on their IP address. If it is able to do this, it will be allowed to continue to participated in the private P2P network. The peer is recognized by its node ID, a string selected by each individual peer.

Peers only connect to peers they have met on the LAN. To increase the number of available resources in the network, peers will forward the search to all its connected peers. In this way, peers who are searching for content will have access to the content of its peers, and their peers' peers.

[7] Tsne is the system which will be used as a basis for the implementation. It is described in greater detail in the next chapter, and even more technical details are discussed in the design and implementation chapters.

The greatest feature of the Tsne network is its autonomy. Moments after the application is started for the first time, the peer is automatically connected to all peers on its LAN. Over time the network builds lists of locations for peers that are not always connected directly on the LAN but rather from remote IP addresses on other locations, and even track dynamic addresses that do not change too often quite effectively. This autonomy is a disadvantage when it comes to keeping the network private.

There is no authentication in Tsne. Because of this the private P2P network is very susceptible to spoofing. An attacker spoofing a node ID will be granted access to all resources in the network . All peers and even an attacker who is able to listen in on the traffic (which is always unencrypted) going to a single node, will be able to listen in on searches, and retrieve the node IDs of all currently connected nodes (which may later be used for spoofing). In addition to this, the resources are made available by a separate port opened by Tsne. Tsne runs a web server on this port, and will serve any shared resource to anyone who connects. The formatting requests for resources is well known, and an attacker may easily gain access by requesting files identified by a numeric ID starting at 0 and incremented every time new content is added by the sharing peer.

Finally there is no mechanism for grading access. All peers are either part of the network or not. Peers in the network are granted full access to searching and files.

4.5. Specifying the challenges

As the review of current private P2P networks shows, there are some general challenges related to grading of the security. First of all, nodes need to be authenticated. With proper authentication, a network has more information to use for grading security, for example grading access.

As part of authentication there has to be a mechanism for exchanging keys. As we see, common solutions use out of band channels or concepts like web of trust or adaptations of this. All of these approaches has its greatest challenge when exchanging keys for the first time, especially when a new invited peer is joining the network. It will be attempted to find a suitable mechanism for exchanging a key, and for associating it with an identity. Trust is an important aspect of authentication, so it should be looked into how trust may be used in the setting of key exchange and authentication.

When a peer is authenticated, access may be graded based on many parameters. It is evident that peers using private P2P networks wants to restrict access to resources for one reason or the other. It will be attempted to find a suitable solution for grading access to resources. In this context grading access could generally be interpreted as granting different peers access to different shared files. However it will be attempted to make the solution suitable for resources on a general level, not only files.

The impression of the author is that all the current private P2P networks has problems with finding a solution which retains the favorable autonomy of a regular P2P network, while providing the control of a private P2P network. The private P2P networks seems to either have a very strict control and a low autonomy, or little control (or is at least very vulnerable to attacks) and high autonomy. It will be attempted to find a solution which provides strict control while keeping the autonomy more intact than in current private P2P technologies, and thus merge the two P2P concepts closer together, keeping the best of both and maybe even adding some features.

--

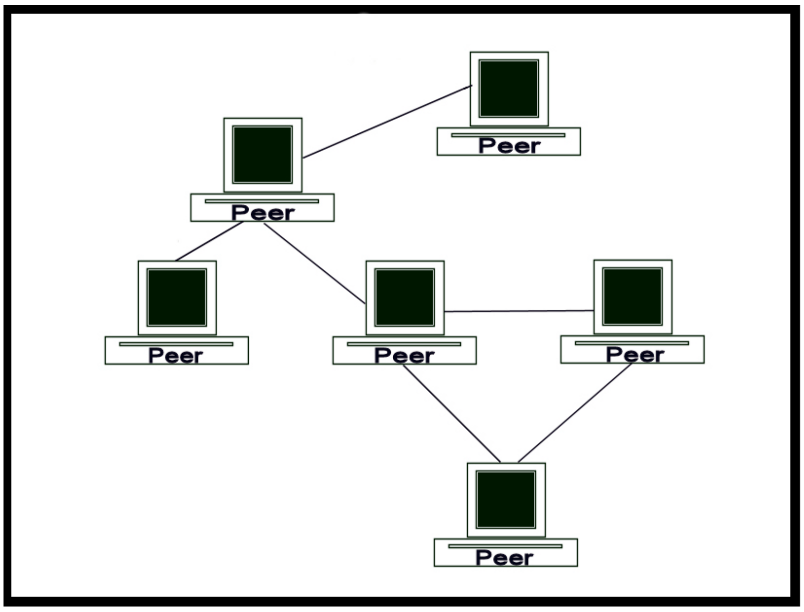
5. The Sozialized.net Embedded – An existing private peer-to-peer solution with limited levels of security

5.1. The Sozialized.net Embedded (Tsne) as a starting point

As previously stated, the implementation of Tsne was going to be used as an example of a private P2P network without differentiated levels of security. This implementation was to be explored, and based on the research in the previous chapters, a restructured design would be developed. The goal of the restructured design would be to introduce mechanisms for differentiated levels of security in the Tsne system, preferably retaining much of the autonomy of the original design. In this chapter we describe the design of the original Tsne private P2P network. This chapter is based mainly on inspection of the source code of Tsne, snooping of traffic which the Tsne network transmits, experimentation with the application, and the insight that came with making the revised version as part of the developed test system. Resources[21,22] written by its creator, Njål Borch, has also been very helpful in researching its operation.

5.2. Overview of Tsne

Tsne is a P2P file sharing application which allows peers to share mp3 files. On the networking and file sharing level the system operates autonomously, or in other words, Tsne discovers peers and adds them to the private network without the interaction of the users of Tsne. Tsne does not rely on a centralized nor a decentralized authority to allow peers to connect to a network or to each other. Instead a Tsne peer builds a network of peers it meets on different LAN networks it is connected to.



--

Figure 3: Example of topology of a private P2P network without a centralized server

Figure 3 shows a typical topology of a small Tsne network. The peers stores the IP addresses of the peers they interact with. If these IP addresses are publicly accessible addresses, a peer that migrates to another network is able to reconnect with a peer that is still on the LAN where they met. The new IP address of the migrated peer (in addition to the IP address it recently had on the LAN) is stored in the peer that did not move. If the unmoved peer also moves at this time, it is able to connect to the peer that moved first, because it already knows its new IP address. As this process continues, the peers continually build a private network which maps different selections of IP addresses as possible locations of given peers. When a node responds to a probe at a given IP address, it is considered active (in contrast to the alternative state of being offline). An active node may then be used by the peer to search for mp3 files. The active node will return a compiled list of URL references to mp3 files on the active node itself and on all nodes the active node has currently registered as active. The searching peer will compile its own list of URL references, this list consists of mp3 files the searching peer it self is sharing, and the responses from all its active peers. This compiled list of URLs to mp3 resources is then passed to the peers media player, and the media player will begin to play the mp3 by streaming the files. Just to clarify it is noted that the files that are being streamed, are located on any of these locations:

- The searching peer itself
- An active peer that has responded with a compiled list of URLs and the URL that is being streamed is located on this active peer's system
- A node that is not known to the searching peer, but an active peer that is known to the searching peer has returned a compiled list of URLs, and the URL that is being streamed references this unknown peer.

The streaming itself is handled by a simple web server which is implemented in Tsne. The web server has no restrictions and will serve any shared[8] resource to any entity which requests it. There is no control of which peers are allowed to connect to the private network or to the web server. In addition there is no control over which of the shared resources the peers (or other probing entities) get access to. In this way, any online entity which is capable of following the

[8] The web server selects which file to serve by parsing the request looking for a resource identification. This identification is used to do a look up in the web servers database. The database contains a mapping between resource identifications and shared files. The web server will thus only serve files which are identified by a resource identification.

http protocol could download any file from any peer in the private network by simply probing the http servers and requesting random resource identifications.

5.3. The inner workings of the original Tsne

In this section we take a closer look at the inner workings of the original Tsne. Based on the research in the previous chapters and our examination of Tsne we will use the next chapter to identify sections of the code where it may be possible to introduce some of the different levels of security we have discussed. In the next chapter we will consider which of these are suitable for alterations or additions without changing the original idea of Tsne. Or in other words, we will attempt to introduce levels of security while keeping the autonomy of the system as intact as possible.

5.3.1. Announce – peer announcing its presence (neighbor discovery)

The basic operation of a private P2P network is a mechanism for discovering and registering peers. In Tsne this is handled by socializing with nodes on a LAN and the process is as follows:

> To start Tsne, tsne.py is executed without parameters. By doing so, all peers running Tsne uses the module neighbord (neighbor discovery) to open a server socket on a predefined port and starts listening for incoming network traffic. We will call this part of the system for the daemon.
>
> All Tsne peers will periodically broadcast[9] an Announce[10] message to this same port, announcing their presence on the LAN, their random user identification (ID) and the implied intention to participate as a peer.
>
> Any Tsne peer which receives this broadcast announcement will check the user ID against its database to see if this is a new peer or a peer it has met before. If it has not seen it before the new peer is added to the database with the source IP address of the broadcast, if it has seen the ID before the source IP address of the broadcast is added as another possible address of this ID.

[9] Tsne was actually written supporting IPv6 link local multicasts, the equivalent of IPv4 broadcasts. The restructured design uses IPv4 broadcasts, and thus the two will not be differentiated in this setting.

[10] The word Announce is the first line of the message (just like GET is the first line of a http request) and indicates to Tsne which subroutine should be called to handle the incoming message.

This means that any Tsne peer on a LAN network or migrating between different LAN networks, for example at work, a university, a WLAN hot spot or similar, will socialize with other Tsne peers and gradually populate its database with a list of IDs and associated IP addresses.

5.3.2. Ping Pong – Keeping track of active peers at 'any' location

The database of peers a given peer has meet will continue to grow each time it meets a new peer in this way. To maintain a list of peers that are available when a given peer connects to a network, a pinging scheme is used:

> All peers running Tsne uses the module neighbord (neighbor discovery) to open a socket on a predefined port and starts listening for incoming network traffic (the same port that listens for the announcements mentioned above)
>
> All peers will continuously check the database entries for each of the peers they have met and examine the timestamp of their last contact with any given peer.
>
> - A very recent timestamp indicates an active peer
> - A recent but older than a few seconds indicates an active but stale peer.
> - A timestamp which is older than approximately one minute indicates an offline peer.
>
> Stale and offline peers will be sent a ping. Stale peers are pinged on the IP address they are currently active on, while offline peers are pinged on all known previous IP addresses. When a pong is received the timestamp will be updated (and thus the peer will be considered active again). If a given peer does not respond with a pong, the timestamp will eventually expire (or the timestamp is already expired), and the peer is considered offline.

When a peer receives a ping with the ID of a peer it has meet before, the source IP address of this ping is added to the database as a possible address of this ID (if it is different than the previously stored IP addresses of this ID). If we look at two peers, and both peers leave the LAN at the same time, they will not be able to reconnect until they are both on the LAN at the same time again. However, if one of the peers move to a new IP address (at home, an office or similar) and pings the peer which has remained on the LAN, the LAN peer will then know two possible addresses of the peer that moved. If the LAN peer then moves to another IP address, it will attempt to ping the peer that moved first on both its known locations. Given that the first peer has not moved again, receiving a ping with the ID of a peer it has seen before, will update the database with another location to connect to the peer with this ID. If the networks are suitable the peers will still be able to connect[11] even though they have both moved to new IP addresses. As the peers move around, it will become more and more likely that a peer, which moves to a new IP address, will have the current IP address of the peer it is trying to ping among its list of previously known IP addresses for this peer.

5.3.3. File searching and sharing

The private P2P network ,with its announcements, pings and pongs, is the backbone of Tsne. The objective of Tsne is to share resources, specifically mp3 files[12]. Mp3 files are located through a distributed search among the peers, and searched files are available for streaming to anyone who is participating in the private network (or pretending to participate, but more on this later).

Tsne handles searches and streaming of resources through a built in web server which supports a sub set of the http specification. The details are as follows:

> When Tsne is executed without parameters, it uses the module network (web server) to start a web server (in addition to the neighbord daemon explained in the previous section) on a given port.
>
> A user executes Tsne with '-- play' as a parameter, followed by one or several search terms. The search terms would typically be one or several artists, song titles, music genres[13] or similar. For example

[11] Connectivity is a complicated issue. Tsne relies on traffic flowing in both directions directly between two peers. If one of the peers is behind a firewall or a router which does not allow the incoming traffic to pass or does not allow server sockets, one peer may see the other as active, while the other peer sees the first as offline. Similar problems will be observed if one of the peers are on a network which uses NAT when assigning IP addresses to network adapters that are present on the LAN. Solutions for this problem are considered beyond the scope of this thesis and relates to the original design of Tsne and its intended use.

[12] Tsne is programmed to share mp3 files, but it is noted that rewriting Tsne to share any resource that may be represented as a read only stream is by the author of this paper considered trivial (this does not imply anything about the amount of time it would take to perform the actual rewrite). Because Tsne operation is based on searching, the streams should have meaningful search parameters associated with them.

[13] During testing of the original Tsne it was noticed that searches seemed to only return resources where some or all search terms matched tokens in the file names of resources. Tsne's database, which is used to do the search, is designed to create a mapping between given resources and searchable tokens in the file name, but also between given resources and tokens in the ID3 tag for the given resource. The searchability of the ID3 tag is considered insignificant for this thesis and thus the reason for this problem was not explored. It is however noted that Tsne either has a bug in its handling or searching of the ID3 tag, or the test system had a compatibility problem with the original Tsne code. This problem carries over to

--

```
tsnecv.py --play Morten Abel Inparticular
```

Tsne then sends a http query request to the local instance of the Tsne web daemon, with the search terms as part of the query.

The query originating from the peer itself is handled in the same way a query from any other peer would be handled. The web server performs a search among the local resources and compiles a list of URL references which contains its own IP address and the assigned resource ID of all (if any) resources which match the search keywords. The web daemon then passes the same query to all active peers listed in its database (and because all peers are running Tsne this is repeated throughout the private P2P network). This results in a flooded distributed search, and the local web server will add all timely search responses to its list of URLs that references resources which match the search terms. Finally, the local web daemon will return a http response with its list to Tsne (which as we remember was called with --play as a parameter and is not to be confused with the Tsne daemons that are running the Tsne system itself).

Tsne will store the transmitted list and then execute the operating system's default media player or the mpeg123 media player if the environment variable for media player isn't configured. When executing the media player, path to the stored compiled list of URL references is passed as a parameter.

The media player interprets this as a regular play list (which it in fact is), and begins to stream the mp3 resources like it would any other resource. The media player sends a http get request to the given IP addresses in the play list, with the resource ID where one would 'usually' see a file name (i.e. http://158.39.41.256:3074/resource?id=1329 [14]).

The respective network daemons on the peers that are referenced will then respond by streaming the resource their database has associated with the resource ID.

the implementation of the revised Tsne, and thus the revised version is also only able to search the file names of resources, not the ID3 tag. This may or may not be the case for any other test system which experiment with the revised version.

[14] Note that identical resource IDs has different associations on different peers. Also note that this example IP address is non existent.

--

5.4. Levels of security in the original Tsne

In the next chapter we will explore the levels of security that Tsne does not have and the levels that based on our research are considered desirable to attempt to introduce. This is a large topic and is more suitable for discussion in the design of the revised version of Tsne. In this short section however, we will briefly mention the few levels of implied security that Tsne already has and a few obvious problems related to this.

In Tsne any peer which is present on a LAN network with another peer is autonomously added to the list of peers. Peers has no means of accepting or rejecting these added peers. It is possible to clear all peers but not just one single peer (even though this would be trivial to implement). The problem is however, that even if a peer is deleted it will immediately be added again if it is still on the same LAN, or even if it ever sends a ping to an IP address that the deleting peer is at.

Resources on a Tsne system are either shared or not shared. Typically one or several folders containing mp3s are imported into Tsne and made available for searching and streaming. Any peer which is active will have access to all shared resources (actually even a hostile peer pretending to be offline would have the same access, but more on this in the next chapter).

The implied security that we associate with Tsne is that users on a LAN may be peers that other peers on the same LAN choose to trust somewhat more than random peers on other locations on the Internet. There are many problems related to this implied security and they will be discussed and attempted to be resolved in the revised version. A few of the related problems are as follows:

> If Tsne is used on a laptop, it will typically intermittently be connected to both LANs with peers one would choose to trust and LANs with complete strangers, and there is no mechanism for differentiating between these.
>
> If there were a mechanism for differentiating between peers(for example a trivial revision allowing manual approval or disapproval of the addition of peers) there would still be no way to authenticate an approved peer connecting remotely at another time.
>
> Even on a LAN with peers one would normally trust, there is no mechanism for excluding or limiting access to resources for known hostile peers, nor is there a mechanism for granting selected peers access to specific restricted resources.

6. Specification and design

6.1. The Sosialized.Net Embedded - Cryptographic Version, an envisioned private peer-to-peer solution with graded levels of security

As part of the research in this thesis, Tsne was revised to support grading of security. In the following section the specification of the expected security of the revised version is specified. The specification was used to create a design for the revised version, The Sosialized.Net Embedded - Cryptographic Version (Tsnecv). This design was the basis for the implementation of Tsnecv, described in chapter 7 of this thesis. The design shows the features the system should be able to simulate. In the implementation some of the features are handled directly by the system, while others are handled by manipulating the database entries for peers in the system.

6.2. Specification of Tsnecv

This section specifies the conditions that defines the expected operation and security of Tsnecv. All the conditions had to be fulfilled to achieve what is considered a secure system. The consideration of secure only applies to the level of security that is implicitly provided by the specification.. Truly Secure Cryptographic systems are large and complicated, and are only as strong as their weakest link. The focus of this thesis was grading of security, and the features that were selected were selected to achieve or demonstrate that such grading is possible, not to make the system as a whole completely secure from every possible attack. As an example, the research that has been done has described known cryptographic systems which has shown that if entities are authenticated, it is also possible to create a secure channel. This realization was used to select authentication as a feature of the specification. In a truly secure cryptographic system the authenticated peer would use a secure channel to retrieve a file, and the file would thus only be readable by the authenticated peer. According to the following specification we choose to ignore that the file must be accessed on a secure channel, and only require that a peer must be authenticated before access is granted to the file. This allowed us to focus on how the authentication and other aspects of trust might be used to grade security by giving selective access to files. According to the specification the implementation is what is considered to be secure in the setting of this thesis. This does not however imply that the system as a whole is truly secure in a real world setting.

6.2.1. Specification

The following specification of the systems behavior aimed to autonomously but cryptographically secure grant access to any new peers, while allowing peers to share some (if any) files with all peers and selected files with selected groups of peers (where a group may possibly contain just 1 peer).

--

The application shall be able to execute on a Windows XP 32 bit computer system with a LAN adapter, connected to a LAN, where other instances of the application, executing on other Windows XP 32 bit computers, may join and leave at their own discretion.

The network shall function while connected to the Internet, not just in a test laboratory.

Peers must be authenticated in a cryptographic secure manner before they are allowed to participate in the private network.

If a peer is authenticated and is believed to reside at a given IP address, it is not considered a breach of security if an entity uses the unsecured channel of the web server (which every peer makes available to other peers through Tsnecv) to gain access to resources by spoofing the given IP or acting as a man in the middle making Tsnecv believe that the authenticated peer resides on the attackers IP address.

A session key which could be used to create a secure channel shall be exchanged during authentication. This key should be exchanged to demonstrate that the secure channel is available even though it is not used when downloading files.

The quality of external cryptographic libraries shall not be considered a part of the systems level of security.

All peers on a LAN shall be able to join the network autonomously.

All peers shall have the option of making selected resources available to all peers.

All new peers shall be able to autonomously provide their key to existing peers by in band communication

All peers shall have the option of obtaining or confirming keys of other peers through a mutual peer which of both trust the public key completely, if such a mutual peer exist.

All peers shall have the option of manually confirming keys and then to increase their trust in the key associated with the peer.

All peers shall have the option of manually increasing their trust in the key associated with a peer without manually confirming the key.

Tsnecv shall autonomously assign trust to a new peer in the network based on how the public key of the peer is obtained.

Tsnecv shall autonomously give authenticated peers access to files shared to specific groups of peers based on how much trust is associated with the key identifying the authenticated peer.

--

All peers shall have the option of making selected resources available to only a selected group, of which at least one group should be used to represent the peers a single person is the owner of. The number of separate groups is not important, but it should be at least 5, and a group should be able to contain zero or more peers.

6.3. Design of Tsnecv

If we look back at chapter four where suitable technologies were discussed, the following aspects of security were identified:

Authentication (origin Integrity) - Which entity is communicating

Data integrity - Is the data unchanged

Confidentiality (access) - restricted access to data, or at least restricted access in the form of only selected entities being able to decrypt data that is visible to all(meaningless to all others)

Secrecy - Is the communication invisible to others

Availability - Is the data available when an entity requires access

When we designed a revised version of Tsne we had to consider which of these were desirable for the intended purpose of Tsne. We also had to consider which of these might be implemented without interfering more than necessary with the autonomy which is a founding concept in the original design. Another very important aspect was how strong the security should be. By strong we mean how much resources, be it computational power, bandwidth, storage or other, we were willing to use on a specific level of security. For example, looking back at authentication we remember that increased key length for a given cryptographic algorithm generally increases the security for the algorithm. But, generating longer keys require more computational cycles, and decryption and encryption with longer keys also require more computational cycles. We also note that a longer key requires more space to be stored and to be transmitted.

6.3.1. Selected levels of security

Confidentiality in the form of access control and origin integrity in the form of authentication has been the focus of most of this research. The research show that these are levels of security that are considered desirable by most of the private P2P networks. For the intended private use of Tsne they are definitely considered to increase the security level of the network.

--

6.3.2. Selected encryption algorithms and bit strength

6.3.2.1. *Symmetric encryption with AES*

For algorithms in Tsnecv which rely on symmetric encryption, the leading standard AES was selected. The algorithm has been determined to be secure by a broad base of researchers, some of these have been referenced in chapter four. 3DES and Serpent were considered, but AES was selected both because of its standardization and its presence in the elected cryptographic package.

6.3.2.2. *Asymmetric encryption, signatures and authentication with RSA*

For algorithms in Tsnecv which relies on asymmetric encryption, the leading standard RSA was selected. The algorithm has been determined to be secure by a broad base of researchers, some of these have been referenced in chapter four. Elliptic Curve Cryptography was also considered due to its use of shorter keys to achieve the same bit strength as RSA. There are two reasons why ECC was not selected. First of all, there was no readily available or renowned implementation of ECC with could be incorporated directly with Python. Implementing 'just' ECC for python is by the author considered far beyond the scope of this thesis . The second reason was that RSA is, even though it is very secure, based on mathematical concepts which are covered by the mandatory mathematical courses leading up to this thesis. As a result the author is able to understand how it works, and thus reduce the risk of using it incorrectly (which is a serious threat in all cryptographic systems).

6.3.2.3. *Selected bit strength of RSA and AES*

According to RSA Laboratories[24], 1024 bits is currently considered sufficient for corporate (including online transactions) use, but the security is expected to be breached in the not so distant future. A bit length of at least 1024 bits should be used, but if testing determines that a regular home computer is able to efficiently use stronger keys (e.g. 2048 or 3072)without dramatically impacting performance, stronger keys should be used. It must be noted that there should be a correlation between strength of the symmetric keys used and the RSA keys, in order for the increase to truly increase the security as much as intended . The following table shows the relationship between RSA bit length and symmetric bit strength according to RSA Laboratories.

RSA 1024	Symmetric 80 bit
RSA 2048	Symmetric 112 bit
RSA 3072	Symmetric 128 bit

Table 1 - Correlation of RSA key bit lengths and bit security of symmetric block algorithms

--

6.3.3. *Encryption module*

An external library which implements the selected cryptographic algorithms was used. The following modules were designed to encapsulate access to the cryptographic functions:

6.3.3.1. crypformat

Helper functions which converts tuples of strings to one single string which may then be encrypted and decrypted (by another module), and converted back to the same tuple in a predictable fashion.

6.3.3.2. crypglobals

A static file which only contains variables used to configure the operation of the cryptographic system.

6.3.3.3. crypdocrypto

Helper functions which was designed to perform the actual cryptographic functions by using a publicly available cryptographic implementation of RSA and AES. If the selected cryptographic algorithms are exchanged for similar ones, this should be the only module that must be rewritten extensively (even though its format should not need to change).

6.3.4. *Authentication and key exchange module*

The existing module in Tsne neighbord (neighbor discovery) is in Tsnecv responsible for initiating authentication and key exchange based on expired or non existent authentication state for a connecting peer. The actual work is carried out by the protocol part of the module.

6.3.4.1. neighbord (neighbor discovery)

The operation of the original neighbord module is specified in chapter 5.3.1 and 5.3.2. The revised version in Tsnecv was designed to uses a scheme similar to the existing Ping - Pong scheme used for discovery. Just like there is a timeout for nodes becoming stale or offline, the redesigned version was designed to use a timeout for the authentication process. The design specifies that Authenticated nodes be registrerd as AUTHACTIVE. When the timeout (typically configured to be 60 seconds) expires an authenticated node should be flagged as AUTHSTALE, an soon after it should be flagged as AUTHEXPIRED. Active and stale peer(in terms of Ping - Pong) which are not AUTHACTIVE should be authenticated again. This solution was selected because access to the web server does not use a secure channel. Because a peer may be contacted by anyone on the last known IP address a peer has connected to, it was necessary to combine the authentication with a timeout.

--

Just like Pinging, this could be viewed as an implementation of presence, ie actively polling peers to confirm they are still connected. The only difference is that the 'authentication ping' includes a mechanism for authentication. To perform the authentication the design specifies that a crypprotocol object is to be instantiated in a manner that represents an authentication protocol.

6.3.4.2. crypprotocol

Class that defines variables needed to perform all the protocols. One object of this class should be created every time a protocol is initiated. It was designed to use separate protocol objects for every connection to another peer where either an authentication or key exchange was initiated, and even one separate protocol object for each protocol it was involved in with the same peer. The progress through the protocol was designed to be controlled by repeatedly requesting each object to continue to the next step of the protocol if possible after checking the internal state of the protocol object.

6.3.4.3. crypprotocolgetunsecurekey

The algorithm was designed to handle the protocol for the unsecure public exchange of a relatively untrusted key. The exchange design allowed in band exchange over an unsecure channel.

6.3.4.4. crypprotocolgetsecurekey

The algorithm was designed to handle the protocol for the secure exchange of a relatively trusted key through a common trusted peer.

6.3.4.5. crypprotocolauth

The algorithm was designed to handle the protocol for the authentication of any peer, using the known public key of the peer (whether the key is trusted or not). A key is always obtainable from a peer that is attempting to connect (because it may be retrieved unsecurely). Being authenticated does not imply that a peer is identified, it only establishes a relation between a public key and an assumed identity, which may be used in future communications to identify this assumed identity. The trust Tsnecv or the peer has in this key and assumed identity grades the security of the authentication from nearly zero, to complete trust and absolute security equal to that of the encryption algorithms used and the correctness of the implementations of these algorithms and the Tsnecv protocols.

6.3.4.6. crypprotocolspinner

This algortihm was designed as a very simple helper function which should continuously poll to check if a message has been received from the network, and then use crypselecter to determine which protocol object the message is designated for.

6.3.4.7. crypselecter

This algorithm was designed as a helper function which should examine incoming authentication and key exchange communication to determine which protocol object is responsible for handling the communication, based on the source IP address of the communication.

6.3.5. Access module

As Tsne was an existing system, access was not handled as a separate module. Access was designed to be handled by querying stored information about authenticated nodes. These queries are performed in the existing web server module by accessing the database module. The only helper function the design relied on was crypauth.

6.3.5.1. crypauth

Helper functions designed to access stored information about the system, to determine which groups a peer is associated with, and which specific resources these groups has access to.

6.3.5.2. network (web server)

The operation of the original network module is specified in chap 5.3.. The revised version in Tsnecv was designed to enforce the specified access policies which will be described in section 6.3.5.3 (next section) and section 6.3.6. As stated in the specification for Tsnecv, a secure channel is not required to access resources on the web server. The web server in the original Tsne does not limit access to the web server to peers in the network. Because of this, no mechanism was in place to handle access restrictions in the revised version. The revised version limits access in two ways. The web server has been redesigned to use the IP address of the request as an identifier. The web server is designed to only allow access to resources to authenticated nodes (as maintained in the data base by the neighbor discovery). In addition, the web server was designed to use the crypauth helper function to determine which groups the authenticated peer was associated with. Based on the permissions of the groups the peer is a member of, access is graded. A peer is only allowed access to resources which at least one of its groups has been granted access to.

As an additional feature, the web server was designed to provide a resource other than the mp3 files supported by the original Tsne. A special request to the web server allows a peer to request the host machine of the web server to perform a regular search as a peer, and play the resulting play list directly on the host machine. Peers are supposed to only add peers they control (or maybe household members) to the two special groups which are allowed to obtain remote access to this functionality. This was done mainly to illustrate one use of graded security with respect to access in a peer to peer network, beyond that of limiting access in a file sharing private P2P network.

6.3.6. Trust for autonomously granting access

The proposed solution for autonomously granting access in Tsnecv was is to grade the trust of a public key and its associated user ID. The graded trust was then used to grade access to resources according to the autonomously assigned trust. The assignment of trust was done by the neighbord discovery, and enforced by the web server.

The system was designed to assign trust in the following way:

A peer connecting for the first time, which is only able to provide its public key and alleged identity by in band communication should be assigned a trust level of 1, the lowest trust possible (other than 0 which isn't used, but would imply that the connecting peer is not even using Tsnecv but rather performing a port scan or some other form of probing). A Tsnecv network consisting of such peers only, would represent a regular P2P network, not a private P2P network. Peers with trust level 1 were autonomously granted access to files shared to group 1. Adding resources to group 1 is done at Bob's discretion, so Bob does not have to participate in this sharing of resources. The peers of this type is considered anonymous peers.

A key with trust level one may be upgraded to a trusted key by a peer. The upgrade is done at the peer's discretion. This means that the peer has the choice of either manually confirming the received key by out of band communication with a friend. The peer may also select to use his knowledge and compare the supplied identity of a connecting peer with information about a friend. The peer must then decide if it trusts the network and the exchange of the key. For example, if Bob and Alice are sitting in a room, connected to each other through a switch, Bob will observe someone who is identifying themselves as something like 'CoolAlice<and a cryptographic secure random number>'. The probability of someone acting as man in the middle to forge the key in this setting would usually be considered minimal[15] , and Bob may

[15] This is a consideration that must be made by the user. Two regular peers connected to a switch on a LAN they trust, are very unlikely to be under attack. If however these two peers were likely to be subjected to industrial espionage or maybe even peers sharing military defense information, extraordinary precautions would have to be taken. These peers would have to consider the possibility of secret agents infiltrating the building, exchanging the switch with a specially designed switch under their control, which would allow them remote access and the possibility to change the content of messages, while acting as man in the middle. This indicates that security is relative. For the average peer, considerations such as switches and routers with firmware with security holes are a greater threat, as this may allow hackers to perform the same operation as those described with the secret agent (and secret agents may of course also use hackers)

select to upgrade this key to trusted. To manually trust the key implies that Bob knows Alice, so upgrading the key also upgrades Alice to become a friend. Friends with confirmed keys are autonomously granted access to resources shared to group 7, 5, 3 and 1.

Bob also has the option of manually upgrading an assumed ‘key to identity’ binding for Alice to become a friend without upgrading his trust in the key. Such an upgrade implies that Bob is quite certain it really is Alice, but until it is confirmed he is not quite as willing to share resources as he will be when the binding is confirmed (by himself or other peers). The manual upgrade to friend but with untrusted key autonomously grants the friend access to resources shared to group 5, 3 and 1. If Tsnecv is at a later time able to have the key confirmed by another friend with a secure connection, the key is upgraded to trusted and the friend is granted the additional access of resources shared with group 7 (so total access is 7, 5, 3 and 1)

Friends of friends (FoF) are handled in two ways. If Alice is a friend of Bob, and Bob has a trusted key to Alice, Tsnecv trusts Alice’s judgment and autonomously grants FoFs that reach Bob through Alice access to files shared to group 3 and 1. Adding resources to group 3 is done at Bob’s discretion, so Bob does not need to rely in the judgement of his friends even though Tsnecv does so. If Bob does not have a trusted key to Alice, Tsnecv autonomously treats the FoFs who reach him through Alice the same way he treats anonymous peers.

It is noted that if the friend of a friend with an unconfirmed key connects to the same LAN as Bob, this friend of friend would autonomously be treated as a peer connecting for the first time, and thus be autonomously granted access to the resources of group 1.

The first time Tsnecv is started, it adds itself to group 9. This is a special reserved group which has access to the resources of all other groups. There is another reserved group, group 8. Peers may only be added manually to group 8 and 9 (except for the described addition to group 9 when the system is initialized for the first time). Group 8, like 9, has access to the resources of all other groups. Group 8 is reserved for what should be considered a house hold or similar. The group is used to share a special resource, the ability to invoke remote playing of mp3 files. In a typical configuration, a peer will add all its devices to group 9 and have access to private files shared among its devices in group 9. Household members, such as a media PC in the living room is added to group 8, allowing different members of a household to share music to group 8 and invoke the remote play function of the media PC in the living room from any peer they have added to the media players group 8. The media player is not able to play music shared to group 9. The media PC operates as any other peer in the system, so the members of the household must use the Tsnecv running on it to add their peers to the media players group 8.

--

The special groups with negative numbers, -9 through -1 could be used by a peer to make specific files available to a specific group of peers. The members of each group could be added manually by the sharing peer from the list of available peers in the network. These groups would be intended for friends, and would require members to be friends with trusted keys.

Information used for autonomous grading of peer access based on characteristics of friendship and key to identity binding	**Autonomous Group access**
FRIEND with confirmed/trusted key	**7**
FRIEND with unconfirmed/untrusted key	**5**
FoF through FRIEND with confirmed/trusted key	**3**
Anonymous peer with any key	**1**
FoF through FRIEND with unconfirmed/untrusted key	**1**
All others (probing, protocol errors etc)	**0**

Figure 2: Table of autonomous access. FoF is a friend of a friend, FRIEND is a friend. Group 0 implies no access.

6.3.7. Important manual operations in Tsnecv

The following operation was designed to configure Tsnecv

- Select a nick name which will be recognizable by friends.

The following operations were designed to allow peers to control the managment of their trust in other peers.

- Upgrade to friend by trusting key.
- Upgrade to friend but do not trust key.

  Add peer to group 8 or 9.
- Delete peer to reset all info, but allow to reconnect as a new peer.

--

7. Implementation

7.1. Implementing The Sosialized.net Embedded - Cryptographic Version

In this chapter the actual implementation of Tsnecv, the revised version of Tsne, is described. It will show which mechanisms are used to fulfill the requirements of the specification of the required security stated in chapter 6 of this thesis. It will be demonstrated that the requirements were met, or if the attempt to implement a specific feature was not successful the problems leading to failure will be explored.

The algorithms will be discussed in general terms, for details the reader will be referred to the source code which is a part of the deliverables for this thesis.

7.2. Preparation, tools, System software and hardware setup

7.2.1.1. Hardware

Because it was planned to use a Windows Mobile Wlan capable phone, it was decided to use Windows as the development platform. To make sure the system would really work as described, in the real world, a setup resembling a typical system (except for the few number of nodes) was used. 2 machines were connected to a switch. These machines were connected to the Internet through the Internet line of the switchand were assigned public IP addressed. A laptop was connected to the Internet line from a completely different network segment (two separate Internet providers, HiTos/Uninett and Bluecom). As both lines were available in the same room, the real life situation of a peer moving to a new network could be tested by plugging the network cable back and forth between the switch or the separate ADSL line. The Bluecom line was completely open (no firewalls or NAT), but the Uninett line had a very limited selection of ports which were open for incoming traffic, so the Tsnecv configuration files were set to use these open ports.

7.2.1.2. Software

The development was done in XEmacs. The application that was to be furter developed, Tsne, was available in Python, and Tsnecv was thus naturally developed in Python . Two Sourceforge projects were used: PYsqlite was used as the database, and pYcrypto[25,26] was used by algorithms which had use for cryptographic functions. pYcrypto is a quite low level interface and implementation which mainly contains the ciphering mechanisms and key generation. Functionality like padding, Cipher Block Chaining (CBC)[6,27], and signature verifications had to be implemented separately. The binaries for these two projects are a part of project's code delivery, along with the source code of both Tsne and the developed Tsnecv

--

7.3. Overview of the important modules of Tsnecv

The Tsne code was well structured, but communication in the neighbor discovery was based on a 'fire and forget' concept. A peer would send a Ping and forget all about it. Later it would probably receive a Pong, and update a timestamp for the last contact with whoever was pinged. This approach is a good implementation of a ping, but complicates cryptographic protocols. Authentication and key exchange is usually done by sending several messages back and forth. This is usually done in a connected mode where the two parties maintain a channel throughout the process.

Because the neighbor discovery uses connectionless communication it was decided to use the same for the authentication protocols. This was done in order to follow the original design of Tsne, even though additional steps would have to be taken. Several helper functions had to be implemented in order to resolve issues related to the connectionless operation. The main issues were to maintain and update the state of each step in the authentication or key exchange, while directing incoming protocol messages to the object which was involved with exactly that message. At a single point in time, Tsnecv may have several active protocol objects. For example, 20 objects which are authenticating 20 different peers and a few objects which are attempting to exchange keys. This will lead to at least the same number of messages, but they will arrive in no particular order, on the same port. While waiting for responses, the objects must also handle timeouts, and resends.

7.3.1. Encryption module

The encryption is very important for the secure operation of the system. Because of this the cryptographic module was fully implemented. It used the pYcrypto library for the actual encryption. This library includes the actual encryption, decryption, signing, signature verification, and key generation algorithms. However, the framework for a cryptographic system is much more than these functions. In the following we look at the most important classes in the module.

crypdocrypto formats all calls to the actual cryptographic functions. It performs functions such as padding, unpadding, conversion of data to a form that may be encrypted, the CBC mode used with AES, and a few other formatting functions. It also incorporated testing functions which were used to verify the correctness of the cryptography.

crypauth was not very sophisticated, but provides a solution which may be used to easily associate folders with groups of peers that should have access to them. The folders holds an indicator of required access level in their path name, and the folders may be used at both one or several levels of depth.

crypglobals only holds configuration parameters, but is mentioned because it allows tweaking of the protocols, particularly the behavior which is taken when keys are missing, not trusted and so on. It also allows tweaking of the strength of the cryptographic algorithms used in pYcrypto

7.3.2. Authentication and key exchange module

As stated earlier, the use of a ping pong scheme in Tsne's neighbord daemon, with connectionless communication, made cryptographic protocols a challenge. This was solved by creating a spinning algorithm which would continuously wake protocol objects which still hadn't completed their task.

The protocol object is a derived class from the auth, getunsecurekey and getsecurekey classes. This allows the protocol objects to hold all state for the protocol which is carried out. An alteration to the neigbord daemon implemented a spinning fucntion which continuously woke protocol objects and allowed them to update timers and state and decide if they had timed out, should retry, or if they had used to many attempts to without succeeding (leading to failure). The spinner in the neighbord deamon used a similar scheme to pinging to determine when a node should be authenticated.

The idea was that a an authenticated node should be allowed access to files for a period of time (one minute) after each authentication. When this was about to expire, the node would be authenticated again, allowing it to continued access to the system.

The individual protocol classes auth, getunsecurekey and getsecurekey had the responsibility of defining the steps of each protocol. Get unsecure key was implemented completely using real cryptographic keys from the encryption module. In situations where a protocol would fail, the object would self destruct. It would then be the responsibility of the neighbord daemon to spin a new object after a given time had passed.

auth implemented the steps in the protocol, and would correctly abort if it did not have the required key to authenticate. The authentication was simulated without the use of real kesy, and rather simulated by manipulating the database entries or by inserting errors in the code of the test system .

The implementation of get secure key was eventually abandoned, due to an unresolved problem with the signature verification. A manual copy of a signed text, transported by a removable storage device would work, but when transmitted over the network, the signature verification would always fail.

--

7.3.3. Access module

There was no access restrictions to searching. The enforcement of graded access was handled by the network class. Based on either combinations of key sources, status as a friend, friend of friend, the trust of the key source, or manipulated values of these values access to files was either granted or denied. These variables represented a combination of how much the peer trusted whoever was connecting, and how much the peer trusted the link between a public key and it's assumed identity.

The required user level (or group) was determined by requesting it from the crypauth class in the encryption module.

In addition the network module would confirm that the authentication and key exchange module currently considered the connecting peer to be authenticated.

If the access level of the user was sufficient the requested files would be served. These files were served unencrypted. If the peer was not authenticated or did not have the proper authentication, the protocol would simply abort and not give a response in order to avoid leaking information. This is considered a good cryptographic practice. When something fails, everything one was working on should be shredded.

In addition to files, the network class also makes remote play available. By requesting a particular url, the the same procedures as with file access is followed, but only users with the two highest user levels which are currently authenticated are granted access. If access is granted, the peer will begin to play music locally, requested to do so by a remote peer. This access was designed to be available for the peers own nodes in the same system, and also for household members if so desired.

7.4. Daily use, and testing

7.4.1. Daily use

The Tsnecv network has been continuously tested throughout it's development. One peer has generally been used to store music, and a laptop on the other side of the room, connected to a different network, has been used to play music with remote play. Two friends of the author has been allowed to test the system during its development, and it has behaved as expected according to the stage it was at, at the time of testing. At that point and with it's given configuration it was able to support remote play, secured by the knowledge of a secret key. At this point remote play is fully supported with the exchange of keys and by adding the peer to the restricted user groups (either 8 for household or 9 for the owner). The system is also capable of granting peers access to files based on which user level they have. Peers who join the network are all granted access to the lowest group autonomously

--

7.4.2. Testing

7.4.2.1. Authentication

when peer x is trying to authenticate peer y, the authentication will fail if peer y does not have peer x's public key. The related protocol object will revert to public key exchange. At the next attempt to authenticate, peer x will authenticate peer y. Signatures are not tested or created by invoking them from the encryption module, but the protocol follows the procedure. Increase of trust may be simulated by upgrading peers to friends or household members by accessing the database. This results in increase access

7.4.2.2. File streaming and Remote play access

The file streaming access is fully implemented. Based on the granted access and a current authentication, only files which are supposed to be available are made available. If a peer leaves it's IP address, a new peer joining on the same address will have the possibility to probe for files for up to 60 seconds, due to the timeout of the authentication, and the unencrypted channel that is used for transfers

7.5. Limitations and problems

The system does not use signatures, but the algorithms are implemented in the encryption module.

The trusted key exchange was abandoned due to the problems with the signatures.

Access is not secure with respect to the unsecure channel that is used when transferring files.

The problems were not directly related to the protocols but rather to the complicated nature of cryptographic systems. Debugging is extremely difficult to perform, due to the inherent secret content of what is being debugged. The problem is that it is not always clear at what point of a long chain of events that something goes wrong. It is considered probable that signatures failed because messages were not properly formatted.

7.6. Results and possibilities and further development

The research that was done found some new ideas to grade security. The next generation of the system should be able to implement the signatures properly, and thus authenticate securely. If the transport channel of the web server is encrypted according to an industry standard, it should be possible to make the system quite secure. The grading of access proved to be very exiting. Peer who joined the network by appearing on a LAN, would in matter of seconds gain access to the files made available to the lowest group. Still they would not be able to invoke remote play or gain access to private files. As part of further development, a detailed study of the encryption protocols should be done

8. Conclusions

It has been demonstrated that it is possible to grade security autonomously by assigning trust to the way a key is exchanged. If a peer joins and simply shouts his key, he gets the lowest access. If the key is manually decided to represent his assumed identity, he may be upgraded to access more files. Only if a peer wants to grant more access, does he need to take some form of action.

It has not been proved that the security of the system is intact. Due to the failure of the signatures to verify authenticity , a complete chain of trust is not reached, and the system may be breached by an attacker.

The concept seems promising. The system works and performs many of the operations it was designed to do. The different levels of trust assigned to keys and the ability to accept a received key based on information about who is on the same network at that time, without manually checking the key itself, may be found sufficient for grading security.

--

9. References

1. Rogers, M, Bhatti, S: “How to Disappear Completely: A Survey of Private Peer-to-Peer Networks”.

Paper at SPACE 2007 (Sustaining Privacy in Autonomous Collaborative Environments). Available at http://www.cs.ucl.ac.uk/staff/mrogers/private-p2p.pdf (2007)

2. Wikipedia, “Distributed Transient Network”. Article.

Available at http://en.wikipedia.org/wiki/Distributed_Transient_Network

3. Wikipedia, “Ad-hoc”. Article.

Available at http://en.wikipedia.org/wiki/Ad-hoc#Ad_hoc_computer_network

4. Wikipedia, “Brute force attack”. Article.

Available at http://en.wikipedia.org/wiki/Brute_force_attack

5. Bishop, M, Computer Security: Art and Science, Addison-Wesley, Boston, MA (2003)

6. Ferguson, N, Schneier, B, Practical Cryptography, Wiley Publishing, Inc., Indianapolis, IN (2003)

7. Johnsen, B, Kryptografi: En prosjektorientert introduksjon, Tapir Akademisk Forlag, Trondheim (2005)

8. Wikipedia, “Direct Connect Network”. Article.

Available at http://en.wikipedia.org/wiki/Direct_Connect_network

9. NeoModus, “Direct Connect”, website (now closed) .

--

Available through mirroring service, The Wayback Machine at http://web.archive.org/web/20050612080703/http://www.neo-modus.com/

10 . “DC++”, website.

Available at http://dcplusplus.sourceforge.net/

11. “YnHub”, website.

Available at http://www.ynhub.org/

12. “Octopod”, website

Available at http://sysnet.ucsd.edu/octopod/

13. Balakrishnan, H, Kaashoek,F, Karger,D,Morris,R,Stoica,I, :“Looking up data in P2P Systems”, Communications of the ACM (February 2003)

Available at http://www.project-iris.net/irisbib/papers/dht:cacm03/paper.pdf

14 “Open Direct Connect”, website.

Available at http://sourceforge.net/docman/?group_id=36589

15. “WASTE”, website.

Available at http://waste.sourceforge.net/

16. Sayers,R: “Secure File Transfer With WASTE “, video.

Available at http://showmedo.com/videos/video?name=sayersWaste000&fromSeriesID=63

17. “Retroshare”, website.

Available at http://retroshare.sourceforge.net

--

18. Popescu, B, Crispo, B, Tanenbaum, A: “Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System”, Proc. 12th Cambridge International Workshop on Security Protocols (April 2004).

Available at http://citeseer.ist.psu.edu/popescu04safe.html

19. “The Free Network Project”, website.

Available at http://freenetproject.org/

20. Clarke, I, Hong, T, Miller, S, Sandberg, O, Wiley,B: “ Protecting Free Expression Online with Freenet”, IEEE Internet Computing article (2002)

Available at http://citeseer.ist.psu.edu/462603.html

21. Borch, N: “Social Peer-to-Peer for Social People”, The International Conference on Internet technologies & applications, Wrexham (September 2005).

Available at http://www.socialized.net/#Papers

22. “The Socialized.Net”, website

Available at http://www.socialized.net/

23. R.Rivest, A.Shamir, L.Adleman: “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”, Communication of the ACM, vol. 21(2), pp. 120-126 (1978).

Available at http://citeseer.ist.psu.edu/rivest78method.html

24. RSA Laboratories :“TWIRL and RSA key size”, article

Available at http://www.rsa.com/rsalabs/node.asp?id=2004

25. Python Cryptography Toolkit, website.

Available at http://www.amk.ca/python/code/crypto

26. Python Cryptography Toolkit Documentation, website

Available at http://www.amk.ca/python/writing/pycrypt/

Or In the source code [25], in the file ‘pycrypto-2.0.1\pycrypto-2.0.1\Doc\pycrypt.tex’

27. Weiss, J, Java Cryptography Extensions: Practical Guide for Programmers, Elsevier Inv. (2004)

--

28. IEEE 802 specification, website

Available at http://standards.ieee.org/getieee802/

29. Kazaa, website

Available at http://en.wikipedia.org/wiki/Kazaa

30. Kazaa, website

Available at www.kazaa.com

31. Bharathidasan, A, Ponduru, V:"Sensor Networks: An Overview", paper, University of California, Davis, CA (August 2002)

32. Gerk, E, : " Overview of Certification Systems: X.509, CA, PGP and SKIP", MCG publications, website.

Available at http://mcwg.org/mcg-mirror/papers.htm

33. Finney, H: "Hal Finney Essays: PGP Web of Trust Misconceptions", website

Available at http://www.finney.org/~hal/web_of_trust.html

34. Certicom, " Online Elliptic Curve Cryptography Tutorial",website

Available at http://www.certicom.com/index.php?action=ecc_tutorial,home